Université de Carthage
Institut National des Sciences Appliquées et Technologiques
Ministère des Etudes Supérieures et de Recherche Scientifique

Smart Device Project :

IoT Based BinBot : BinBot

Field of study :

Computer Networks and Telecommunication

Prepared By :

Ghada Jeddey
Rayen Hamza
Emna Drihem
Yasmine Ferjani

Jury composed of :

Mr Damergi Emir

Année Universitaire : 2024 - 2025

Acknowledgment

We would like to express our deep gratitude to Mr. Emir Damergi, our professor and supervisor, for his guidance throughout the execution of this project. His expertise, availability, and valuable advice greatly contributed to the success of this work.

We also wish to warmly thank the entire team that collaborated on this project. The dedication, commitment, and team spirit of each member were essential in successfully carrying out this initiative. Every individual brought innovative ideas and continuous support, making this experience both enriching and formative.

This project represents not only an important academic milestone but also an opportunity to develop both our technical and interpersonal skills within a collaborative environment.

A huge thank you to the entire team for their collaborative spirit, determination, and the constant efforts that made the successful completion of this project possible.

Table of Contents

General Introduction

T he rapid evolution of IoT technologies has fundamentally reshaped the way we interact with the world around us, turning ordinary objects into intelligent, interconnected systems that communicate seamlessly with one another.
This transformation is not only about convenience and connectivity but also about addressing the growing need for efficiency, automation, and sustainability in various aspects of life. As environmental challenges continue to escalate, particularly with issues such as waste management, energy consumption, and resource depletion, the integration of smart devices has become a critical tool in mitigating these concerns.

The merging of environmental causes with the advancement of smart technology offers a unique opportunity to develop innovative solutions that can reduce human impact on the planet. Smart devices, powered by IoT, provide real-time monitoring, data analysis, and automation that can optimize processes, conserve resources, and promote eco-friendly practices. From energy-efficient homes to waste management systems, these technologies enable more effective resource utilization, waste reduction, and better management of environmental resources. This growing trend highlights the potential of smart devices to not only enhance convenience and improve quality of life but also to play a pivotal role in fostering sustainability and promoting a greener, more connected world.

Project Summary

The idea for this project was inspired by the ongoing challenges in waste management at the university, where complaints about overflowing bins and the difficulty of ensuring timely cleaning are common.
Given the large size of the campus, cleaning staff sometimes overlook certain bins due to the sheer volume they must manage. With this in mind, the project focuses on developing an innovative BinBot solution to address these issues.

The BinBot leverages IoT and AI technologies to optimize waste collection processes, improve hygiene, and promote sustainability. Features such as real-time fill-level monitoring, odor detection, and Alexa integration enhance its efficiency and user-friendliness, while the AI system predicts the need for cleaning and sends alerts when necessary. This solution not only addresses immediate waste management concerns but also offers a scalable approach to improving urban waste management systems more broadly.

Chapter 1 : Preliminary Study

1.1 Needs Target clients:

Waste management is a crucial issue, especially in academic environments like universities, where trash bins are often overloaded or poorly maintained. This creates problems related to hygiene, efficient resource management, and aesthetics. The BinBot project aims to solve these issues by offering an innovative solution that automates and optimizes waste management, whether in universities, schools, businesses, offices, or even public spaces. The unit cost of the hardware components is estimated to be between 80DT and 90DT. The selling price for the target audience would be set between 100DT and 110DT, to remain affordable while covering production and integration costs.et d'intégration.

1.2 Smart Device Presentation :

The BinBot is equipped with multiple features that simplify and optimize waste management. It provides the ability to check whether the bin is full or not while enabling hygienic disposal of waste through automated lid opening and closing. These features, detailed below, showcase how the BinBot combines convenience, modernity, and advanced technology to address waste management needs in various environments.

- **Presence Detection:** An external ultrasonic sensor detects the presence of an object within 5 cm.
- **Automatic Opening and Closing:** If the bin is not full and an object is detected, the lid opens to allow waste disposal

and automatically closes after 5 seconds.

- **Fill Level Monitoring:** An internal ultrasonic sensor measures the fill level and publishes the data on the Blynk application.
- **Opening Restriction When Full:** If the bin is full, the lid does not open unless manually commanded via Alexa or the Blynk application.
- **MQTT Publishing:** The bin's fill level and status are published on the smartbin/status topic.
- **Voice Control via Alexa:** Allows users to check the bin's status and open the lid using the "Open" intent.

Product Name	Quantity	Price	Total price
Ultrasonic Sensors	2	4,950	9,900
Esp32	1	33,800	33,800
BreadBoard	1	34,700	34,700
Servomotor	1	15,500	15,500
MF MM FF Cables	1	3,500	3,500
Total			97,400

Table 1: Product price

Here are some shots we took during testing; the code through the online simulator Wokwi, MQTT server connexion , Blynk interface and the hardware protoype .

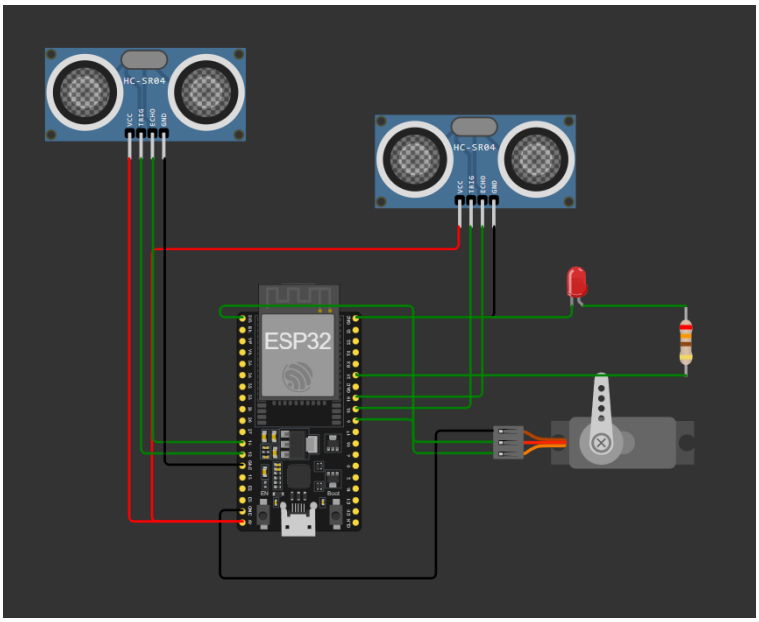

Figure 1: Prototype of the Smart Bin on Wokwi
https://wokwi.com/projects/416233701195029505

Figure 2: Prototype of the Smart Bin

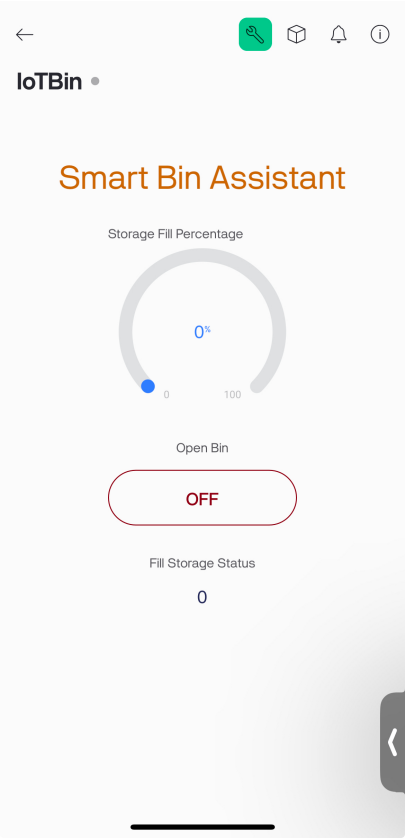

Figure 3: Prototype of the Smart Bin on Blynk

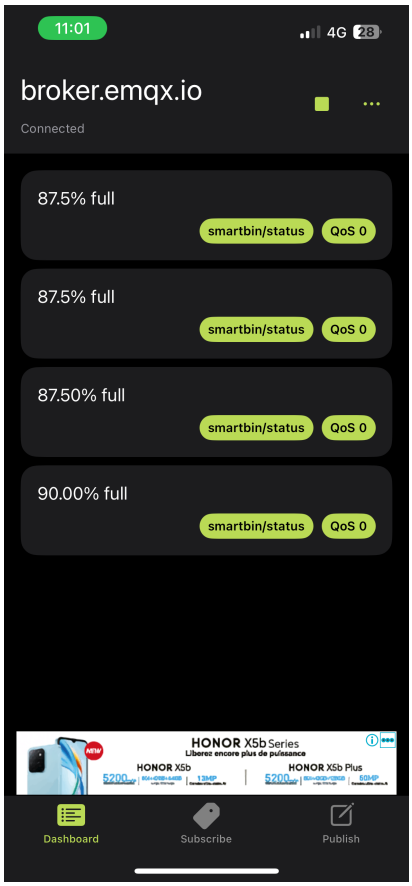

Figure 4: Prototype of the Smart Bin with MQTT Testing

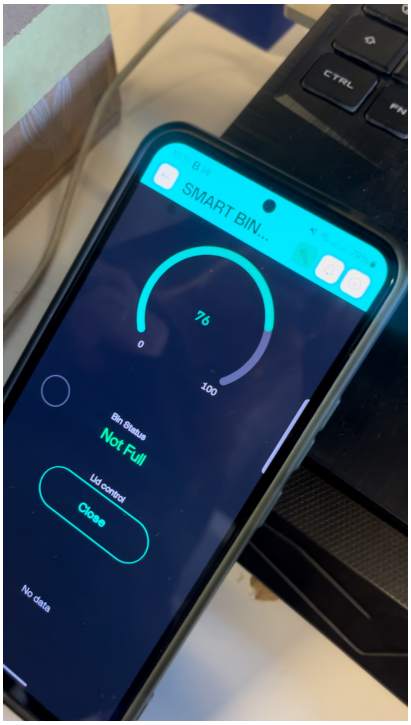

Figure 5: Prototype of the Smart Bin on Blynk

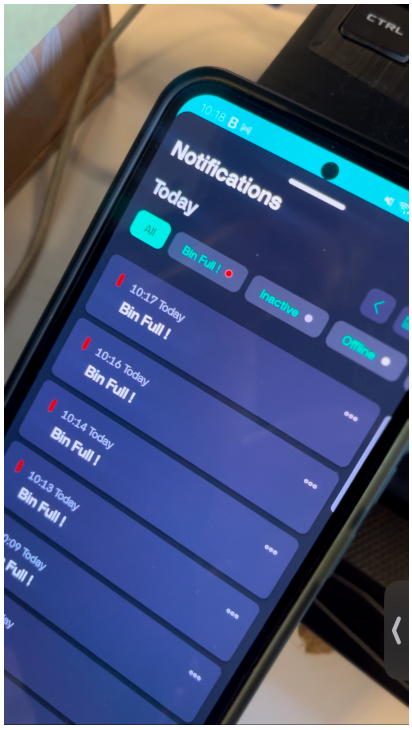

Figure 6: Prototype of the Smart Bin Alert

Chapter 2 : Detailed Study

2.1 Consultation and Update of Variable Parameters

A Smart Device like the BinBot requires several adjustable parameters that can be updated by the end user:

- **Wi-Fi SSID:** The name of the network the bin connects to..
- **Wi-Fi Password:** The key to access the network.
- **Fill Level Threshold:** To set the percentage at which the bin is considered full.

The parameters are saved in JSON format on the ESP32's flash memory. The user can view and modify them through the Blynk application or the Alexa interface.

```
// Param tres variables par defaut
char ssid[] = "DEFAULT_SSID";
char pass[] = "DEFAULT_PASSWORD";
int threshold = 90; // Seuil par d faut (en %)

// Fonction pour sauvegarder les param tres en JSON
void saveParameters() {
  DynamicJsonDocument json(512);
  json["ssid"] = ssid;
  json["pass"] = pass;
  json["threshold"] = threshold;

  File file = SPIFFS.open("/config.json", "w");
  if (file) {
    serializeJson(json, file);
    file.close();
  }
}

// Fonction pour charger les param tres en JSON
void loadParameters() {
  if (!SPIFFS.exists("/config.json")) {
```

```
    saveParameters();
    return;
  }
    File file = SPIFFS.open("/config.json", "r");
  if (file) {
    DynamicJsonDocument json(512);
    deserializeJson(json, file);
    strlcpy(ssid, json["ssid"], sizeof(ssid));
    strlcpy(pass, json["pass"], sizeof(pass));
    threshold = json["threshold"];
    file.close();
  }
}

// Blynk Handler pour mettre    jour les param tres
BLYNK_WRITE(V2) {
  strlcpy(ssid, param.asStr(), sizeof(ssid));
  saveParameters();
}

BLYNK_WRITE(V3) {
  strlcpy(pass, param.asStr(), sizeof(pass));
  saveParameters();
}

BLYNK_WRITE(V4) {
  threshold = param.asInt();
  saveParameters();
}
}
```

2.2 Acquisition et contrôle :

2.2.1 HC-SR04 Ultrasonic sensor

For the BinBot project, we chose the **HC-SR04 ultrasonic sensor** due to its affordability and good performance for small distances (2 cm to 400 cm). Compared to the MaxBotix MB1010, which excels in longer-range performance, the HC-SR04's range is more than sufficient for our needs. Its accuracy is comparable to the JSN-SR04T and better than IR sensors, making it an ideal choice for our application. With a power consumption of just 15mA, it is more efficient than the VL53L0X, though it consumes

slightly more power than the MaxBotix MB1010.

The HC-SR04 is also small in size, which is crucial for fitting inside the bin. While the VL53L0X offers a more compact design, the HC-SR04 strikes the right balance between size and functionality. Although the HC-SR04 is designed for indoor use and is sensitive to noise, it works perfectly in this project. However, for future prototypes intended for outdoor use, we would need to consider sensors that are less susceptible to noise and have better environmental resilience, such as the MaxBotix MB1010 or JSN-SR04T.

Feature	**HC-SR04**	**MaxBotix MB1010**	**JSN-SR04T**	**VL53L0X**	**IR Sensors (Sharp)**
Range	2 cm to 400 cm	20 cm to 7650 cm	20 cm to 600 cm	30 cm to 200 cm	4 cm to 150 cm
Power Consumption	15 mA	9.8 mA	15 mA	20 mA	Varies
Accuracy	High for small ranges	High for long ranges	Similar to HC-SR04	Moderate, sensitive to noise	Low
Features	Affordable, compact, indoor use	Outdoor use, less noise-sensitive, waterproof	Waterproof, outdoor use	Compact, low power, short range	Affected by environmental conditions

Table 2: Comparison of Sensor Features for BinBot Project

This is how we integrated and used the HC-SR04 sensor in our system :

```
#define TRIG_PIN 18            // Trigger pin for the first
   ultrasonic sensor
#define ECHO_PIN 19            // Echo pin for the first
   ultrasonic sensor
#define TRIG_PIN_INSIDE 12     // Trigger pin for the second
   ultrasonic sensor
#define ECHO_PIN_INSIDE 14     // Echo pin for the second
   ultrasonic sensor
```

Listing 1: creating variables for the PINS

```
    long getUltrasonicDistance(int trig, int echo) {
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  return pulseIn(echo, HIGH) * 0.0343 / 2;  // Convert time to
     distance (in cm)
}
```

Listing 2: Function to Measure Distance Using Ultrasonic Sensors

```
    boolean isBinFull() {
  float distance = getUltrasonicDistance(TRIG_PIN_INSIDE,
     ECHO_PIN_INSIDE);
  Serial.print("Distance inside bin: ");
  Serial.println(distance);
  return (distance < 8);  // If distance is less than 8 cm, the
     bin is considered full
}
```

Listing 3: Checking if the Bin is Full

```
    float distance = getUltrasonicDistance(TRIG_PIN, ECHO_PIN);
if (distance < 16 && !isBinFull()) {
  openLid();  // Open the lid if an object is detected near the
     bin and it is not full
} else if (isBinFull()) {
  Serial.println("BIN IS FULL");
  digitalWrite(LED, HIGH);  // Turn on LED to indicate the bin is
      full
  Blynk.virtualWrite(V1, "Full");
  Blynk.logEvent("bin_full__");
} else {
  digitalWrite(LED, LOW);
  Blynk.virtualWrite(V1, "Not Full");
}
```

Listing 4: Monitoring the Bin State (Outside Distance)

```
    void updateBinStatus() {
  float distance = getUltrasonicDistance(TRIG_PIN_INSIDE,
     ECHO_PIN_INSIDE);
  int height = 40;  // Bin height in cm
  float percentage = ((height - distance) / height) * 100;
  percentage = max(0.0f, percentage);

  if (percentage >= 90) {
    digitalWrite(LED, HIGH);  // Turn on LED if bin is 90% full
    Blynk.logEvent("bin_full__");
  }

  // Send status to Blynk and MQTT
```

```
  String statusMessage = String(percentage) + "% full";
  Serial.print("Status Message: ");
  Serial.println(statusMessage);

  client.publish(status_topic, statusMessage.c_str());
  Blynk.virtualWrite(V0, percentage);
}
```

Listing 5: Updating the Bin Status

2.2.2 The SG90 servo-motor

For the BinBot project, we chose the SG90 servo motor due to its affordability and precise control for small angular movements (up to 180°). Compared to stepper motors, which excel in high torque and precise continuous rotation, the SG90's range and accuracy are more than sufficient for controlling the bin lid. Its power efficiency (current consumption of 100–250 mA under load) is better than that of brushed DC motors while offering smoother position control.

The SG90 is also compact and lightweight, which is essential for fitting inside the bin. While micro stepper motors offer a similar compact design, the SG90 provides a better balance between size and functionality for angular adjustments. Additionally, servo motors like the SG90 are easier to program for specific angle movements compared to the continuous rotation of DC motors, making them ideal for simple, precise tasks.

However, for future iterations that might require heavier loads or more complex movements, we might consider gear motors for increased torque or stepper motors for more advanced positioning capabilities.

Below's the code for the configuration and the utilisation of the servo-motor :

```
 Servo myServo;
int pos = 0, speed = 15;
bool lidOpen = false;
```

Listing 6: Servo and state variables

Feature	SG90 Servo Motor	Stepper Motor	Brushed DC Motor	Brushless DC Motor
Precision	Excellent angular control (up to 180°)	High positional control with steps	Low, limited to speed control	Very high speed control, low angular precision
Control Complexity	Simple PWM-based angle control	Requires a driver and step programming	Simple speed and direction control	Requires ESC for control
Torque	Moderate (up to 2.5 kg.cm)	High for larger loads	Moderate, depends on size	High torque and efficiency
Power Consumption	Low (100–250 mA)	Higher due to stepper driver	Varies, generally moderate	High efficiency, low power for equivalent torque
Size & Weight	Small and lightweight	Bulkier for higher torque	Compact and lightweight	Compact but slightly heavier
Durability	Moderate, suited for light applications	High, long lifespan	Low, brushes wear out over time	Very high, no brushes to wear
Cost	Affordable	Expensive	Very affordable	Expensive
Suitability for BinBot	Ideal for precise lid control	Overkill for small bin tasks	Suitable for simple lid operations	Overkill for this use case

Table 3: Comparison of SG90 Servo Motor with Other Motor Types

```
void openLid() {
  if (!lidOpen) {
    while (pos <= 89) {
      pos += 1;
      myServo.write(pos);
      delay(speed);
    }
    lidOpen = true;
    delay(5000);
    closeLid();
  }
}
```

Listing 7: Opening the lid

```
void closeLid() {
  if (lidOpen) {
    while (pos >= 1) {
      pos -= 1;
      myServo.write(pos);
      delay(speed);
    }
    lidOpen = false;
  }
```

```
10 }
```

Listing 8: Closing the lid

2.3 BinBot Processing Overview

For a smart device like the **BinBot**, the primary processing consists of analyzing the data acquired from sensors to make decisions. In the case of the BinBot, the decision-making involves:

Monitoring Bin Fullness

– Using the internal ultrasonic sensor to measure the bin's fill level.

– Determining if the bin is **full** based on a predefined threshold percentage (e.g., 90%).

Automatic Lid Operation

– If the external sensor detects an object within 5 cm and the bin is not full, the lid opens automatically and closes after 5 seconds.

– If the bin is full, the lid remains closed unless overridden by a manual command.

Status Updates

– Communicating the fill level and the bin's status (e.g., *full*, *not full*) to the MQTT broker and Blynk app.

– Triggering a notification if the bin is full.

For a better understanding of the functionalities of the BinBot we present to you the activity diagram :

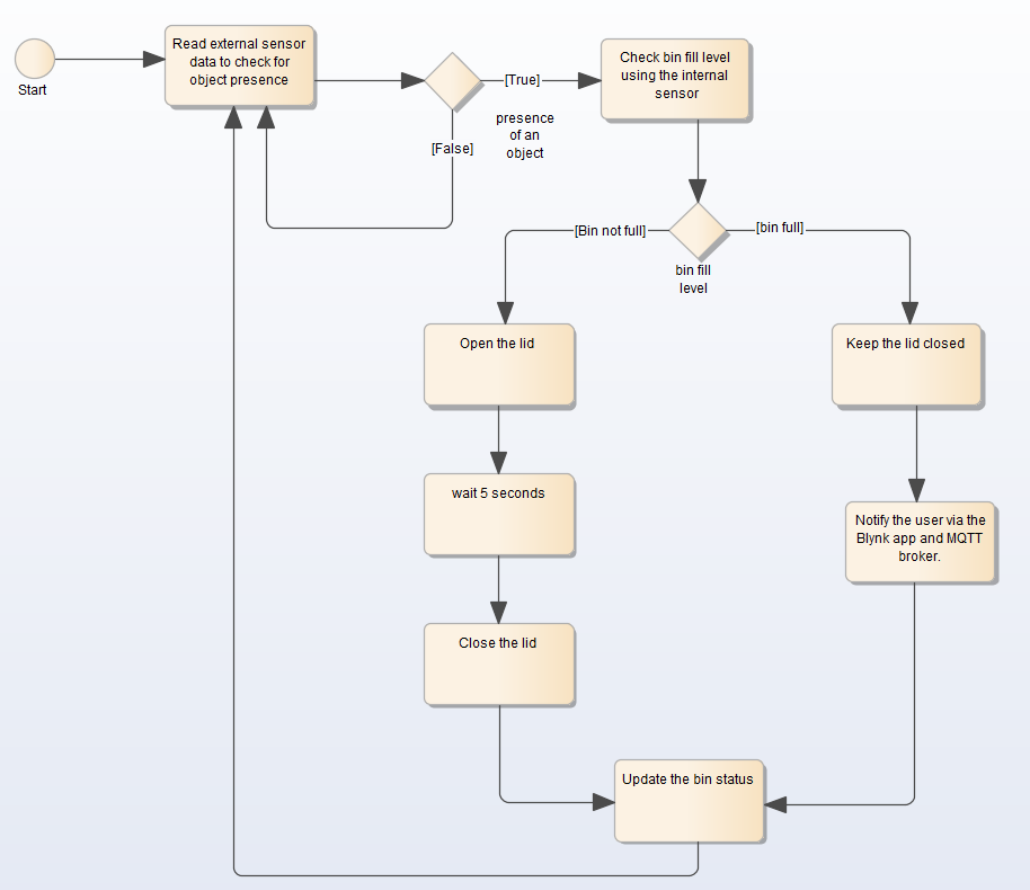

Figure 7: Activity Diagram

```
void loop() {
  // Run necessary services
  Blynk.run();
  timer.run();

  // Reconnect Wi-Fi, Blynk, and MQTT if disconnected
  if (WiFi.status() != WL_CONNECTED) connectToWiFi();
  if (!Blynk.connected()) connectToBlynk();
  if (!client.connected()) connectToMQTT();

  client.loop();

  // Monitor bin state
  float distance = getUltrasonicDistance(TRIG_PIN, ECHO_PIN);
  if (distance < 16 && !isBinFull()) {
    openLid();
  } else if (isBinFull()) {
    Serial.println("BIN IS FULL");
    digitalWrite(LED, HIGH);
    Blynk.virtualWrite(V1, "Full");
    Blynk.logEvent("bin_full__");
  } else {
    digitalWrite(LED, LOW);
    Blynk.virtualWrite(V1, "Not Full");
```

```
  }

  delay(1000);  // Avoid spamming the loop
}

boolean isBinFull() {
  float distance = getUltrasonicDistance(TRIG_PIN_INSIDE,
     ECHO_PIN_INSIDE);
  Serial.print("Distance inside bin: ");
  Serial.println(distance);
  return (distance < 8);
}
```

2.4 Communication :

In this project, communication between the smart device (the BinBot) and external systems is handled using WiFi and MQTT protocols. Here's a breakdown of how they are used:

1. WiFi (Wireless Communication)

- The ESP32 microcontroller used in the project connects to a WiFi network to access the internet and communicate with the cloud services (like Blynk and MQTT broker).
- The WiFi credentials are stored in the code, and the ESP32 establishes a connection during the setup phase.
- The Blynk application, which provides a graphical interface for the user to interact with the BinBot, is used over WiFi. This app allows users to monitor the bin's status and control its lid.

2. MQTT Protocol (Message Queuing Telemetry Transport)

- MQTT is used for real-time communication between the BinBot and external devices (such as Alexa or other IoT systems).
- The BinBot is configured to send status updates about the bin's fullness and lid status via MQTT messages. The ESP32 microcontroller subscribes to MQTT topics like smartbin/lid and smartbin/status, allowing external systems to read or send commands to the device.

– The MQTT broker used in this project is broker.emqx.io (a public MQTT broker)

MQTT Topics:

– **smartbin/lid:** Used for sending commands to open or close the lid of the BinBot.

– **smartbin/status:** Used for sending updates on the bin's status

3. Alexa Integration:

– A voice interface is provided through Alexa, which allows users to interact with the BinBot using voice commands.

– The Alexa skill interacts with the BinBot via MQTT. When a user asks Alexa to open the bin, it sends a command to the MQTT broker, which the BinBot subscribes to. The bin then opens its lid.

– The Alexa skill was developed using the Alexa Skills Kit (ASK) for Python, and it communicates with the MQTT broker to send commands to the BinBot.

The Alexa skill is built using the ask sdk core library and handles several intents, such as opening the bin or checking the bin status. Here's an overview of the relevant handlers:

LaunchRequestHandler: Greets the user and attempts to connect to the MQTT broker.

OpenBinIntentHandler: Opens the bin when the user asks Alexa to open it.

CheckBinStatusIntentHandler: Retrieves the current status of the bin (e.g., "full" or "not full") and provides a response to the user. Here's the specific code for the OpenBinIntentHandler and CheckBinStatusIntentHandler:

```
class OpenBinIntentHandler(AbstractRequestHandler):
    """Handler for OpenBinIntent."""
    def can_handle(self, handler_input):
```

```
        return ask_utils.is_intent_name("OpenBinIntent")(
            handler_input)

    def handle(self, handler_input):
        speak_output = "The bin lid is now open. "
        publish_mqtt_msg("open")
        return (
            handler_input.response_builder
                .speak(speak_output)
                .response
        )

class CheckBinStatusIntentHandler(AbstractRequestHandler):
    """Handler for CheckBinStatusIntent."""
    def can_handle(self, handler_input):
        return ask_utils.is_intent_name("CheckBinStatusIntent")(
            handler_input)

    def handle(self, handler_input):
        global latest_status_message
        if latest_status_message:
            speak_output = f"The bin status is: {
                latest_status_message}."
        else:
            speak_output = "I have not received any bin status
                updates yet."
        return (
            handler_input.response_builder
                .speak(speak_output)
                .response
        )
```

2.5 Consumption :

The BinBot's energy consumption is influenced by its components and functionalities, including sensors, actuators, communication interfaces, Alexa integration, and MQTT connectivity.

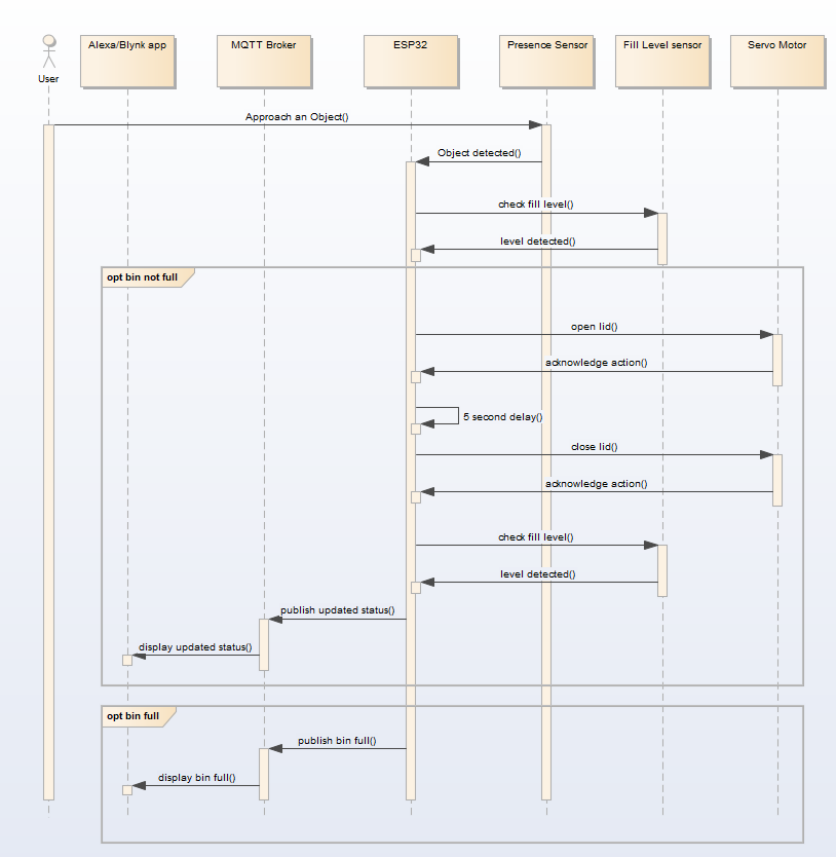

Figure 8: Sequence Diagram of throwing waste

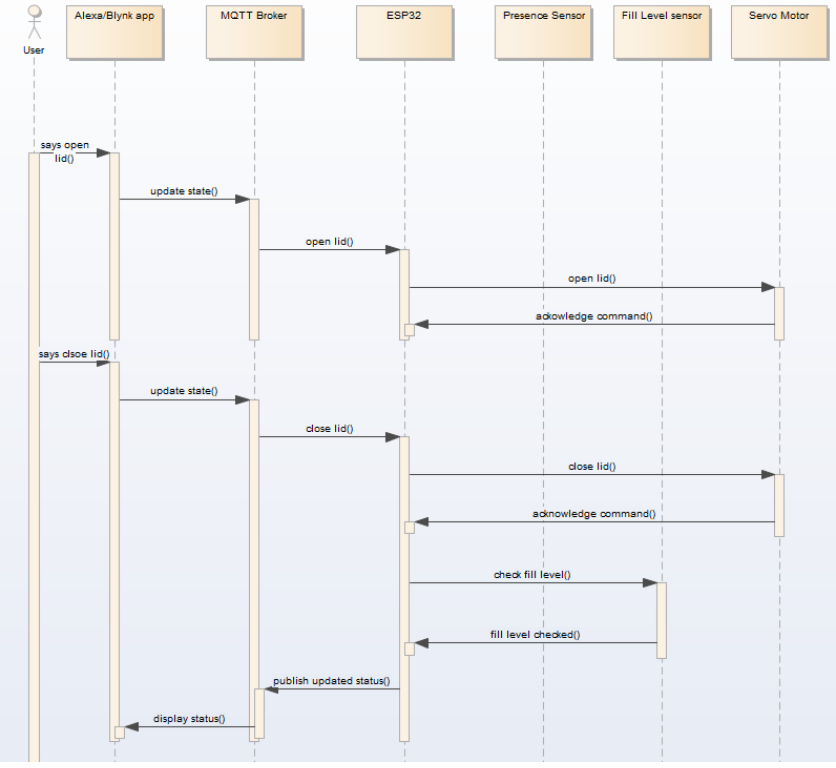

Figure 9: Sequence diagram of cleaning the bin using Alexa

Chapter 3 : Energy Consumption Analysis

3.1 Components' energy consumption : ESP32 Microcontroller

- Average Current Consumption: 80 mA (active Wi-Fi state), 20 mA (idle).
- Operating Voltage: 3.3V.
- Power Consumption:
  - Active: $P = V \times I = 3.3 \times 0.08 = 0.264\,\text{W}$
  - Idle: $P = 3.3 \times 0.02 = 0.066\,\text{W}$

HC-SR04 Ultrasonic Sensors (x2)

- Average Current Consumption: 15 mA each.
- Operating Voltage: 5V.
- Power Consumption (per sensor): $P = 5 \times 0.015 = 0.075\,\text{W}$
- Total Power for Sensors: $2 \times 0.075 = 0.15\,\text{W}$

SG90 Servo Motor

- Average Current Consumption: 150 mA during operation.
- Operating Voltage: 5V.
- Power Consumption: $P = 5 \times 0.15 = 0.75\,\text{W}$
- Note: The servo is typically active only for a few seconds per use, reducing its average power contribution.

Wi-Fi Communication

- Average Current Consumption: 70 mA during transmission.
- Operating Voltage: 3.3V.
- Power Consumption: $P = 3.3 \times 0.07 = 0.231\,\text{W}$

Alexa Integration

- Wake word detection and processing: Additional 20–30 mA.
- Power Consumption: $P = 3.3 \times 0.03 = 0.099\,\text{W}$

MQTT Integration

- Average Current Consumption:
  - Idle MQTT: 10 mA (heartbeat messages to broker)
  - Active MQTT: 20mA (data publishing)
- Power Consumption:
  - Active: $P = 3.3 \times 0.02 = 0.066\,\text{W}$
  - Idle: $P = V \times I = 3.3 \times 0.01 = 0.033\,\text{W}$

Total Energy Consumption Worst-Case Scenario (All components active simultaneously):

$$P_{\text{total}} = P_{\text{ESP32}} + P_{\text{sensors}} + P_{\text{servo}} + P_{\text{Wi-Fi}} + P_{\text{Alexa}} + P_{\text{MQTT}}$$

$$P_{\text{total}} = 0.264 + 0.15 + 0.75 + 0.231 + 0.099 + 0.066 = 1.56\,\text{W}$$

Typical Case (ESP32 idle, sensors active, occasional Wi-Fi and servo activity):

$$P_{\text{total}} = 0.066 + 0.15 + 0.2 + 0.033 + 0.099 = 0.548\,\text{W}$$

Battery Life Estimation (3000 mAh Battery) Battery Capacity in Watt-hours (Wh):

$$E = \text{capacity (Ah)} \times \text{voltage (V)} = 3 \times 3.7 = 11.1\,\text{Wh}$$

Autonomy:

. Worst-Case Scenario:

$$\text{Autonomy} = \frac{11.1}{1.56} \approx 7.12\,\text{hours}$$

. Typical Case:

$$\text{Autonomy} = \frac{11.1}{0.548} \approx 20.25\,\text{hours}$$

To optimize energy consumption, we suggest leveraging ESP32's deep sleep and ULP modes, reducing the frequency of MQTT updates, enabling local wake word detection for Alexa, and scheduling active functionalities during peak usage times.

3.2 Synthesis :

The BinBot's global functioning begins when the user places an object near the bin. The ultrasonic sensor detects the presence of the object. If the bin is not full, the system automatically opens the lid using the servo motor, keeps it open for 5 seconds, and then closes it. Simultaneously, the bin's state is updated and can be monitored through the Blynk app.
If the bin is full, the lid remains closed to prevent overfilling. In this case, the user must use either the MQTT protocol or Alexa to manually open the lid. These commands are processed via the ESP32 microcontroller, which controls the servo motor to open the lid regardless of the bin's fullness status.
This setup ensures both automation and manual control, making the BinBot functional and adaptable to various situations.

Conclusion

In conclusion, the BinBot project represents a significant step towards integrating advanced technologies such as IoT and AI into waste management systems.

By addressing the common issues of overflowing bins and inefficient collection schedules, this solution not only enhances hygiene and convenience but also promotes sustainability.

The innovative features, including fill-level monitoring, odor detection, and AI-driven predictive analytics, enable real-time optimization of waste collection, contributing to more efficient operations and cleaner environments.

With the potential for scalability, this project could serve as a model for smarter, more sustainable waste management solutions in urban settings, offering long-term benefits for both local communities and the environment.